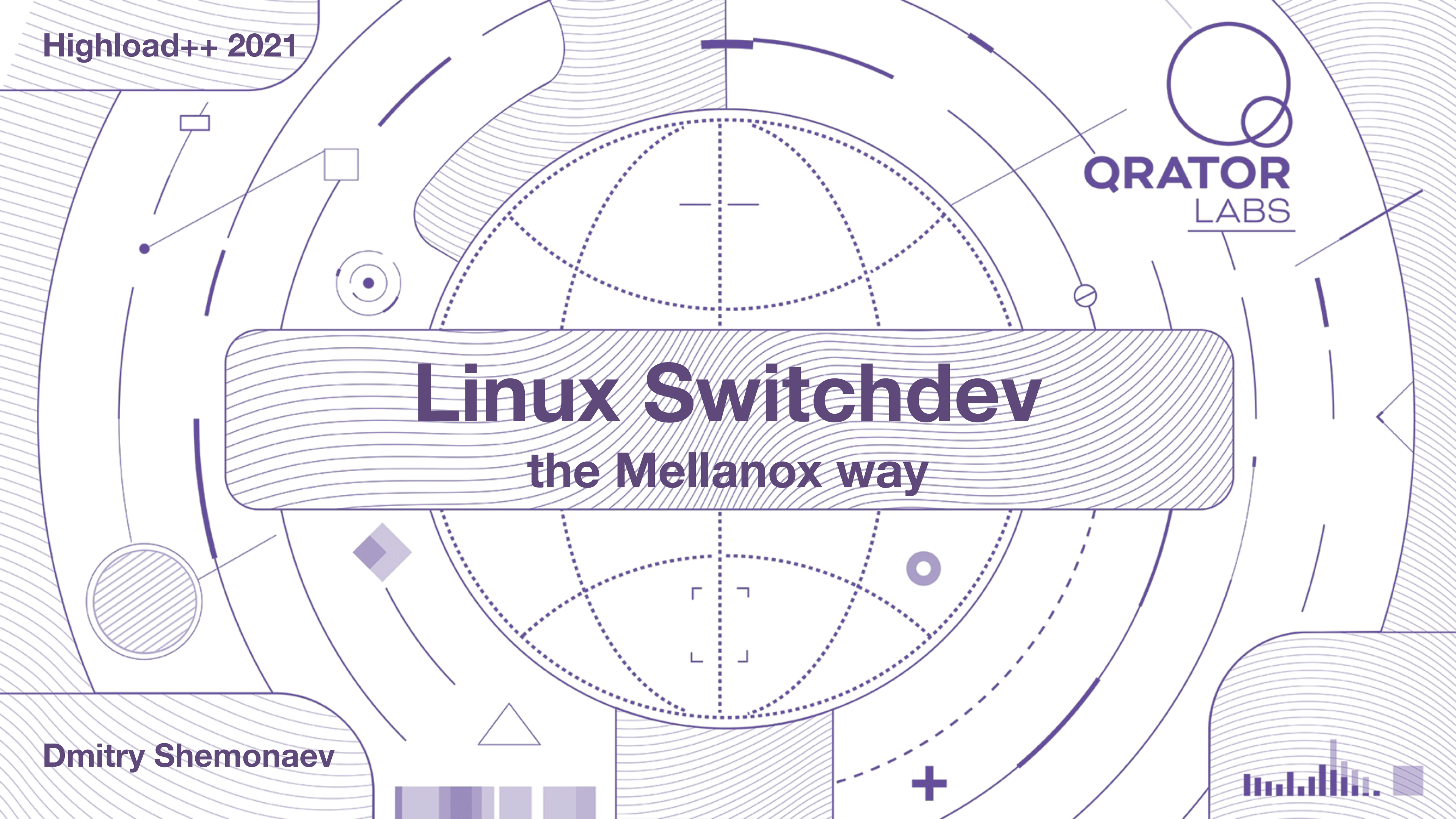

Highload++ 2021
QRATOR
LABS
Linux Switchdev
the Mellanox way
Dmitry Shemonaev

QRATOR LABS

The ways

- Big Fucking Routers
- Switches

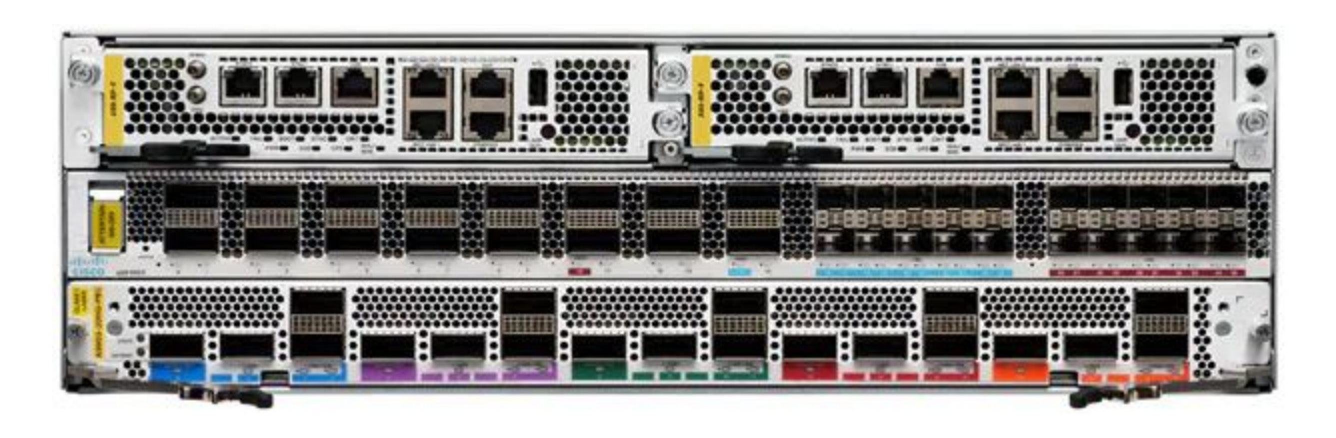

QRATOR LABS


BFR





Already not so big :)





- Big Fucking Routers:
  - Pros:
    - Included hardware HA (NxPSU, NxRE)
    - Full Internet table in FIB
    - Support of various routing protocols (e.g. IS-IS)
    - Support of long range optics (port power out)

- Big Fucking Routers:
  - Cons:
    - Too expensive (and sometimes you must buy a license)
    - Too complicated
    - A lot of power consumption
    - A lot of rack space
    - Not flexible software

Switches



And not so small :)



Switches

- Switches:
  - Pros:
    - Not so expensive
    - Less power consumption (not always)
    - Less rack space (not always)
    - A lot of 100G/400G ports per RU
    - BYOS (Bring your own software, in Linux case)

QRATOR LABS

Switches

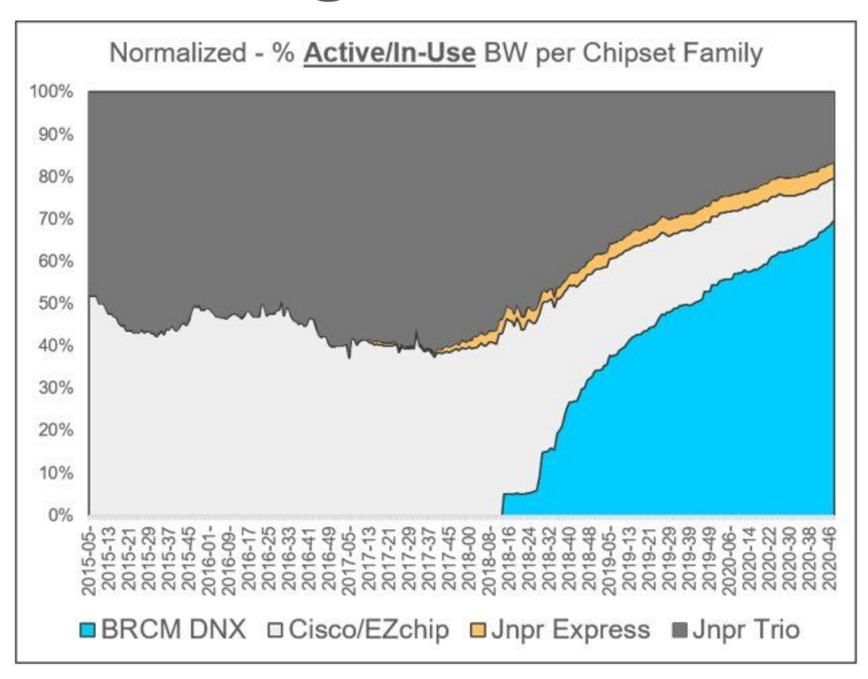
- Switches:
  - Cons:
    - FIB limits (you cannot instal BGP FV in FIB)
    - Fixed hardware configuration (not always)
    - Not reserved RE (not always)
    -
    -

QRATOR LABS

Switches

- Switches types:
  - By hardware:
    - Own chips (Cisco Monticello, Juniper Paradise(Q5))
    - Commodity chips (Broadcom, Intel, Cavium, Barefoot)
  - By software:
    - Proprietary OS (IOS, Junos, EOS)
    - Whitebox (You can change OS)

AS1299 routing silicon evolution



Credits Jonah Gustawsson: https://twitter.com/Gustawsson/status/1353029596661354496



Whitebox switches





- Software
  - MLNX-OS/Mellanox Onyx
  - Cumulus 

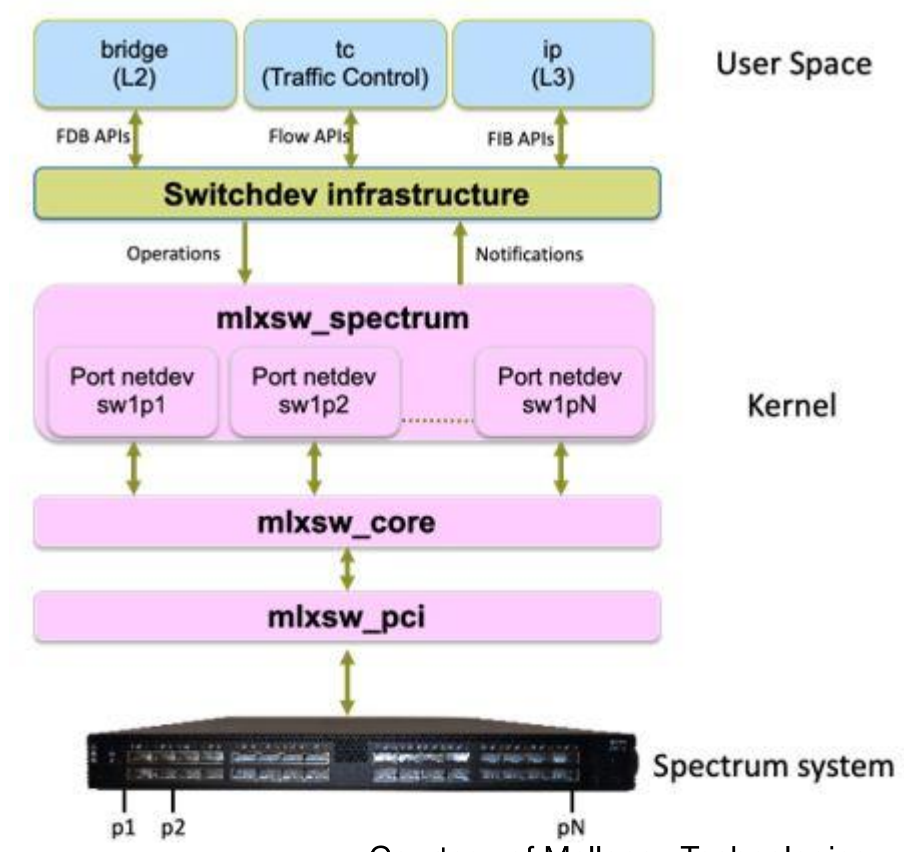
  - SDK
  - SAI (Switch Abstraction Interface), SONiC (NOS)
  - switchdev (Linux kernel)



- https://www.mellanox.com/products/switch-software

Switchdev

- in-kernel infrastructure
- dataplane ↔ Linux (offload)
  - bridging
  - routing
  - filtering
- since 2014
- Mellanox (2015)



Courtesy of Mellanox Technologies
https://blog.mellanox.com/2018/12/mellanox-spectrum-linux-switch-powered-by-switchdev/

Kernel, firmware

- kernel version
  - vanilla (https://github.com/Mellanox/mlxsw/wiki#mlxsw)
  - net-next
- firmware
  - in driver (linux ≥ 4.13, fw ≥ 13.1420.122)
  - tool (mstflint)
- initramfs
  - premature driver load



Linux toolkit

- iproute2
  - ip
  - bridge
  - devlink
  - tc
- ethtool
- lldpad: LLDP, QoS (DCB)
- sysctl: hash policy, qos prio update

QRATOR LABS

We use

- LACP (link aggregation)
- VLAN, bridge (switching)
- VRF (virtual routers)
- ECMP (multipath)
- ACL (filtering)
- GRE (tunneling)
- no PBR (policy based routing)



Configuration way differences

- Monolith configuration vs different configurations files
  - Traditional NOS uses monolith configuration:
    - One big configuration divided by blocks (RP, ACL, System services, Interfaces configuration).
    - ”Syntax sugar”
  - In Linux we have many different config files:
    - You must keep in mind file locations and/or change order
    - sed, awk, grep, ect...



Configuration way differences







Cisco/Juniper LACP example

QRATOR LABS

Linux LACP example



Cisco/Juniper vlan example

Linux vlan example



Cisco/Juniper VRF example



Linux VRF example

- netns / vrf
- special ip rule (v4, v6)

```
1000:    from all lookup [l3mdev-table]
```

- add vrf: link type vrf, vrf ↔ table

```
ip link add name vrf-int type vrf table 200
```

- iface to vrf: ip link set master

```
ip link set dev vlan20 master vrf-int
```

- route between vrfs: explicit dev

```
ip route add 203.0.113.0/24 via 198.51.100.2 dev vlan20 table 100
```



Cisco/Juniper ACL example

QRATOR LABS

Linux ACL example

Configuration

- port → bond → bridge → vlan, loopback → vrf → ip
- restrictions
  - down before set master (port, bond)
  - can not set master to enslaved (bond, bridge)
- init: big script
- runtime changes

QRATORLABS

Configuration

- port → bond → bridge → vla opback → vrf → ip
- restrictions
  - down before set master ond)
  - can not set master to e bond, bridge)
- init: big script
- runtime changes
  - pain

- perl takes care
  - mlxrtr

```
[port 1]
split 4
[bond srv1]
slave port1/0, port1/1
[bond srv2]
slave port1/2, port1/3
[vlan 10]
native port2
vrf ext
ip 192.0.2.2/31
[vlan 20]
tag bond srv1, bond srv2
vrf int
ip 198.51.100.1/24
```

```
[loopback 10]
vrf ext
ip 192.0.2.1/32
[vrf ext]
table 100
route 0.0.0.0/0 via 198.51.100.2 dev vlan20
[vrf int]
table 200
route 0.0.0.0/0 via 192.0.2.3 dev vlan10
route 203.0.113.0/24 via 198.51.100.2 dev vlan20
```



Config example (init, split, link)

```
sysctl -w …
ip rule del pref 0
ip rule add pref 30000 table local
devlink port split pci/0000:01:00.0/25 count 4
tc qdisc add dev enp1s0np1s0 ingress_block 100 ingress
…
ip link add name bond_srv1 type bond lacp_rate fast min_links 1 \
        mode 802.3ad xmit_hash_policy layer3+4
ip link set dev bond_srv1 down
…
ip link add name loop10 type dummy
ip link set dev loop10 down
ip link add name switch type bridge vlan_filtering 1
ip link set dev switch down
ip link add name vrf-ext type vrf table 100
ip link set dev vrf-ext down
ip link add name vrf-int type vrf table 200
ip link set dev vrf-int down
```



Config example (set masters)

```
ip link set dev enp1s0np1s0 down
ip link set dev enp1s0np1s0 master bond_srv1
ip link set dev enp1s0np1s0 down
…
ip link set dev enp1s0np2 master switch
ip link set dev enp1s0np2 down
ip link set dev bond_srv1 master switch
ip link set dev bond_srv1 down
…
ip link set dev loop10 master vrf-ext
ip link set dev loop10 down
ip link add link switch name vlan10 type vlan id 10
ip link set dev vlan10 down
…
ip link set dev vlan10 master vrf-ext
ip link set dev vlan10 down
ip link set dev vlan20 master vrf-int
ip link set dev vlan20 down
```



Config example (vlan, link up)

```
bridge vlan del vid 1 dev bond_srv1
bridge vlan add vid 20 dev bond_srv1
…
bridge vlan add vid 10 dev enp1s0np2 pvid untagged
bridge vlan add vid 10 dev switch self
bridge vlan add vid 20 dev switch self
ip link set dev enp1s0np1s0 up
…
ip link set dev bond_srv1 up
ip link set dev bond_srv2 up
ip link set dev loop10 up
ip link set dev switch up
ip link set dev vlan10 up
ip link set dev vlan20 up
ip link set dev vrf-ext up
ip link set dev vrf-int up
```



Config example (ip, route)

```
ip -4 address add 192.0.2.1/32 dev loop10
ip -4 address add 192.0.2.2/31 dev vlan10
ip -4 address add 198.51.100.1/24 dev vlan20

ip -4 route replace 0.0.0.0/0 metric 0 table 100 proto static \
        nexthop via 198.51.100.2 dev vlan20 weight 1
ip -4 route replace blackhole 0.0.0.0/0 metric 4278198272 \
        table 100 proto static

ip -4 route replace 0.0.0.0/0 metric 0 table 200 proto static \
        nexthop via 192.0.2.3 dev vlan10 weight 1
ip -4 route replace blackhole 0.0.0.0/0 metric 4278198272 \
        table 200 proto static
ip -4 route replace 203.0.113.0/24 metric 0 table 200 \
        proto static nexthop via 198.51.100.2 dev vlan20 weight 1
```

QRATORLABS

Change example

- move port to other bond

```
ip link set dev enp1s0np1s2 down
ip link set dev enp1s0np1s2 nomaster
ip link set dev bond_srv1 down
ip link set dev bond_srv1 nomaster
ip link set dev enp1s0np1s2 master bond_srv1
ip link set dev enp1s0np1s2 down
ip link set dev bond_srv1 master switch
ip link set dev bond_srv1 down
bridge vlan del vid 1 dev bond_srv1
bridge vlan add vid 20 dev bond_srv1
ip link set dev enp1s0np1s2 up
ip link set dev bond_srv1 up
```

- tc (qdisc, filter)
- routed & bridged
- shared acl
  - block (newer tc)
- per-port only
- goto

```
tc qdisc add dev enp1s0np1s0 ingress_block 100 ingress
```

- tc (qdisc, filter)
- routed & bridged
- shared acl
  - block (newer tc)
- per-port only
- goto
- mlxacl
  - chain per vlan
  - chain 0: match vlan

```
tc qdisc add dev enp1s0np1s0 ingress_block 100 ingress
```

```
[vlan10]
ip_proto icmp dst_ip 192.0.2.2 action pass
src_ip 203.0.113.0/24 action drop
dst_ip 203.0.113.0/24 action goto [ex1]
dst_ip 203.0.113.0/24 action drop
action pass
[ex1]
ip_proto icmp action pass
ip_proto tcp action pass
action drop
```

ACL example

```
tc filter add block 100 …

… protocol ip chain 101 pref 1 flower ip_proto icmp action pass
… protocol ip chain 101 pref 2 flower ip_proto tcp action pass
… protocol ip chain 101 pref 3 flower action drop

… protocol ip chain 100 pref 1 flower ip_proto icmp dst_ip 192.0.2.2 action pass
… protocol ip chain 100 pref 2 flower src_ip 203.0.113.0/24 action drop
… protocol ip chain 100 pref 3 flower dst_ip 203.0.113.0/24 action goto chain 101
… protocol ip chain 100 pref 4 flower dst_ip 203.0.113.0/24 action drop
… protocol ip chain 100 pref 5 flower action pass

… protocol 802.1q chain 0 pref 1 flower vlan_id 10 action goto chain 100
… protocol 802.1q chain 0 pref 2 flower action pass
```



mlxacl: apply changes

```
tc filter add block 100 \
  protocol ip chain 102 pref 1 flower src_ip 203.0.113.0/24 action drop
tc filter add block 100 \
  protocol ip chain 102 pref 2 flower dst_ip 203.0.113.0/24 action goto chain 100
tc filter add block 100 \
  protocol ip chain 102 pref 3 flower dst_ip 203.0.113.0/24 action drop
tc filter add block 100 \
  protocol ip chain 102 pref 4 flower action pass

tc filter add block 100 \
  protocol 802.1q chain 0 pref 3 flower vlan_id 10 action goto chain 102
tc filter add block 100 \
  protocol 802.1q chain 0 pref 4 flower action pass

tc filter del block 100 chain 0 pref 1
tc filter del block 100 chain 0 pref 2
tc filter del block 100 chain 101
```



QRATORLABS

```
filter protocol ip pref 1 flower chain 100
filter protocol ip pref 1 flower chain 100 handle 0x1
  eth_type ipv4
  ip_proto icmp
  in_hw
    action order 1: gact action pass
     random type none pass val 0
     index 1 ref 1 bind 1

filter protocol ip pref 2 flower chain 100
filter protocol ip pref 2 flower chain 100 handle 0x1
  eth_type ipv4
  ip_proto tcp
  in_hw
    action order 1: gact action pass
     random type none pass val 0
     index 2 ref 1 bind 1

filter protocol ip pref 3 flower chain 100
filter protocol ip pref 3 flower chain 100 handle 0x1
  eth_type ipv4
  in_hw
    action order 1: gact action drop
…
```

QRATORLABS

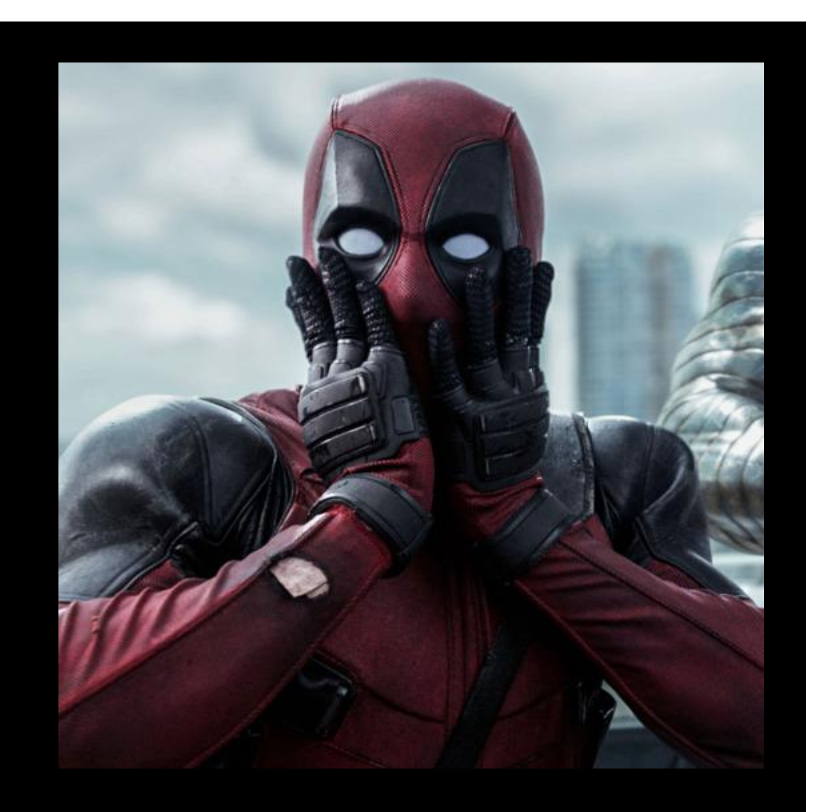


ACL tc filter show

```
filter protocol ip pref 1 flower chain 100
filter protocol ip pref 1 flower chain 100 handle 0x1
  eth_type ipv4
  ip_proto icmp
  in_hw
    action order 1: gact action pass
     random type none pass val 0
     index 1 ref 1 bind 1

filter protocol ip pref 2 flower chain 100
filter protocol ip pref 2 flower chain 100 handle 0x1
  eth_type ipv4
  ip_proto tcp
  in_hw
    action order 1: gact action pass
     random type none pass val 0
     index 2 ref 1 bind 1

filter protocol ip pref 3 flower chain 100
filter protocol ip pref 3 flower chain 100 handle 0x1
  eth_type ipv4
  in_hw
    action order 1: gact action drop
…
```

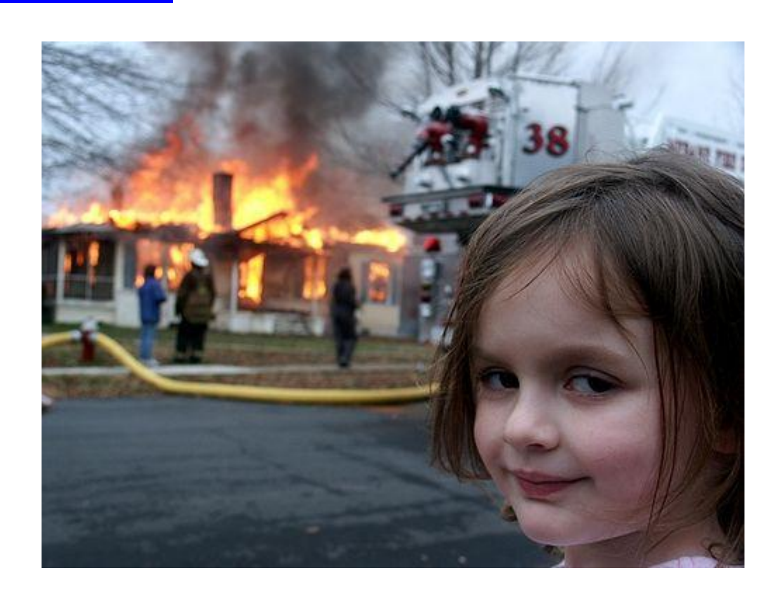


mlxtoolkit

- https://gitlab.com/qratorlabs/mlxtoolkit
- MIT license
- Perl
- mlxacl: 1k lines
- mlxrtr: 2.7k lines
- dependencies:
  - perl modules
  - /root/bin/{bridge,ip,tc}
  - devlink, sysctl



- https://gitlab.com/qratorlabs/mlxtoolkit
- MIT license
- Perl
- mlxacl: 1k lines
- mlxrtr: 2.7k lines
- dependencies:
  - perl modules
  - /root/bin/{bridge,ip,tc}
  - devlink, sysctl



Serial console tips

- ESC, R, ESC, r, ESC, R

Serial console tips

- ESC, R, ESC, r, ESC, R
  - BIOS “Ctrl-Alt-Del”
- SysRq: use “break”
  - minicom: Ctrl+a, Ctrl+f
  - screen: Ctrl+a, Ctrl+b
- BIOS: Ctrl+b

QRATOR LABS

- My contacts
  - Dmitry Shemonaev
  - ds@qrator.net